REMARKS

Reconsideration and allowance are respectfully requested.

Claims 1-22 are patentable under 35 U.S.C. 103(a) over US Patent 5,524,253 (Pham), US Patent 5,297,279 (Bannon), and US Patent 6,640,255 (Snyder) because none of the references discloses the claimed system nor is any reference directed to the problem of sharing objects over a network.

The present invention addresses and provides a solution to a long-standing problem faced by program developers working in object-oriented programming of writing networking instructions that manage the sharing of classes of objects. This is a time-consuming task and involves extensive testing. The present invention describes a unique higher-level programming language that can be used to describe shared objects wherein network instructions are embedded within the object classes. Thus additional network-specific instructions are not needed. The invention relates to object-oriented programming defined in the present claims.

Independent claims 1 and 11 require, inter alia, that a "processing means is configured by said executable instructions set to manage the duplication of said described objects". As described in United States Patent Application No. 09/735,925, which is equivalent to United Kingdom Patent Application No. 00 26 095.0, referenced at paragraph 50 as providing a detailed description of duplicated objects, the management of the duplication of objects involves copying an object to a networked

terminal to create a duplicate and then maintaining consistency between duplicates (see, for example, paragraph 27). None of the references of record including those expressly cited by the Examiner discloses the management of the duplication of objects.

Pham relates to a system for integrating applications that are running on different platforms and written in different languages. According to Pham this problem can arise where processes have been automated one at a time over a period of years resulting in the need to connect machines to each other (column 1, lines 23-42). Pham therefore describes a software tool that "translates" messages and data that need to pass from one machine to another (column 5, lines 11-17).

Pham does not discuss object-oriented programming and thus, Pham does not disclose memory means that "is configured to store program instructions for describing objects to be shared over a network by a plurality of network-connected terminals", nor that "said processing means is configured by said executable instructions set to manage the duplication of said described objects" as uniquely pointed out in claims 1, 11 and 21. Claim 1 defines "objects" to be the elements of object-oriented programming which is not taught nor suggested by the references. Thus claims 1, 11 and 21 are not anticipated by Pham.

With regard to claim 22, Pham clearly does not disclose any of the required elements, such as "object class definition files", "object class description files", "linker", a "Data Definition Language compiler", a "Higher Level Programming

Language compiler", a "Data Definition Language library", or "Higher Level Programming Language libraries".

Pham is directed to the problem of creating a network between differing machines and does not discuss object-oriented programming. The reference is directed to the problem of creating a network between automated systems, which is a totally different problem from that solved by the present invention. Thus Pham neither discloses nor renders obvious the amended independent claims.

Bannon (US 5,297,279) discloses a system and method for database management for providing support for long-term storage and retrieval of objects created by an application program written at least in part in object-oriented programming languages. This type of database is referred to as an object-oriented database (OODB). Previous OODBs have various limitations. Some use new programming languages, some require mapping between data models, some require a proprietary language translator, and so on. Bannon provides an improved system by presenting an application interface for programming languages comprising a number of software modules, using the data model of existing object-oriented languages.

Bannon does not suggest the possibility of object duplication. He is limited solely to storing and retrieving objects, not with duplicating them over a network. Although Bannon discusses the use of a Data Definition Module that "accepts object type descriptions on standard C++ programming

language statements... and extracts sufficient information from the descriptions to enable the OTS module to translate objects between their primary and secondary representations" (column 7, lines 3 to 8), this translation should not be confused with duplication. As described at column 25, line 1 to column 26, line 39, translation between memories moves an object between the internal primary memory and the external secondary memory on a single computer system. It is not the duplication of the object over a network of terminals. Thus Bannon does not disclose memory means that "is configured to store program instructions for describing objects for use in object-oriented programming to be shared over a network by a plurality of network-connected terminals", nor that "said processing means is configured by said executable instructions set to manage the duplication of said described objects".

Snyder relates to a method and apparatus for installing distributed objects on a distributed system. Distributed objects are used within object-oriented programming to provide a specific function. When a programmer requires this functionality he can make a call, through an interface, to the object, without himself understanding the object (column 1, lines 50 to 54). An additional advantage is that the object can be written in a single language but with multiple interfaces, each in a different language, allowing a programmer working in a different language to call an object (column 5, lines 1 to 4).

Snyder addresses the problem that it is difficult to bring a

non-distributed object onto a distributed object system. The problem is solved by including, in the distributed objects, wrapper classes that inherit object attributes through an inheritance relationship with a developer-written class of objects. This means that the programming code required for distributed objects to operate in the distributed object system is provided transparently (column 9, lines 8 - 24).

One of the fundamental differences between Snyder and the present invention is that Snyder is directed to distributed objects, whereas the present application relates to duplicated objects. Duplicated objects are duplicated such that an instance of the object exists on each computer within a network as required. Distributed objects exist in only one location on a network and their functionality is distributed by means of interfaces. Thus these two types of objects are in fact opposite to each other, with the consequence that the problems in using each of them are completely dissimilar.

Thus Snyder does not disclose, as required by claims 1, 11 and 21, that "said processing means is configured by said executable instructions set to manage the duplication of said described objects", because Snyder does not relate to the duplication of objects. Similarly, claim 22 requires that the "program instructions are configured to describe objects to be shared by a plurality of network-connected terminals over a network". Snyder does not disclose the sharing of objects, rather, it relates to access to a single instance of an object.

Thus, the independent claims are novel over Snyder.

Pham (US 5,524,253) does not discuss object-oriented programming and thus does not disclose "program instructions for describing objects for use in object-oriented programming to be shared over a network". The only examples of sharing in Pham are the copying of manipulation files to a compilation node and the distribution of a data manipulator module to a new node during its setup process (see Figure 5, described at columns 13 and 14). Neither of these can be defined as managing the duplication of objects, because consistency between the copy and the original is not maintained. Thus Pham does not disclose the configuration of a processing means by an executable instruction set to manage the duplication of objects.

Snyder (US 6,640,255) discusses distributed objects, i.e. objects which are accessed remotely. Thus Snyder does not disclose that a "processing means is configured by said executable instructions set to manage the duplication of said described objects", since the document does not discuss the duplication of objects but rather accessing a single distributed object from many locations.

Thus none of the documents discloses that a processing means is configured to manage the duplication of objects. Thus no possible combination of the documents could lead a skilled man to suggest the present invention as claimed in claims 1 and 11, since all are missing this limitation.

Additionally, claims 1 and 11 require that the program

instructions (a) describe objects for use in object-oriented programming to be shared over a network by means of ASCII instructions, and (b) compile these ASCII instructions within an executable instructions set that configures the processing means to manage the duplication of said described objects. Similarly, claim 22 requires that the program instructions "describe objects to be shared by a plurality of network-connected terminals over a network by means of ASCII instructions; and compile said ASCII instructions within an instructions set executable by said network-connected terminals." Thus the independent claims require that the ASCII instructions that describe the objects are compiled within an executable instructions set that instructs the processing means to manage the duplication.

In the present application, this is embodied by network instructions being embedded within the object classes (ASCII instructions that describe the objects). Thus, when the classes are compiled, along with other source files, into an executable instructions set, the network instructions are compiled also, as shown in Figure 2. This means that the executable instructions set 208 configures a processing means to manage the duplication of objects. This contrasts with the prior art described with reference to Figure 1, where already-compiled code is altered to implement networking instructions.

However, in Pham the instructions that create the shared files are not compiled into an executable data set. The shared files are created on an administration node (see column 13, line

64 to column 14, line 2), and are then copied to a compilation node (see column 14, lines 41 to 47). The instructions to create them are left on the administration node. Thus even if these files could be considered to be duplicated objects, which as argued above is not the case, then this additional limitation is not disclosed.

Alternatively, the distributed DMM module is created by a DMM builder and C compiler (see column 14, lines 42 to 54). It could therefore be considered as an executable instructions set that contains compiled ASCII instructions. However, it does not configure a processing means to manage the duplication of objects, since there is no disclosure in Pham of duplicated objects.

There are no shared objects in Bannon, and so no disclosure of ASCII instructions that describe objects to be shared over a network.

Snyder does not disclose ASCII instructions that describe objects to be shared over a network as required by the independent claims. Snyder may disclose ASCII instructions that describe objects to be distributed over a network, allowing remote access to an object regardless of its location or language. These instructions are compiled, as described at column 16, line 48 to column 17, line 32. However, the results of the compilation are object files, object servers and client programs - not an executable instructions set. These cannot be used to configure a processing means to distribute the object.

Snyder instead achieves this by linking compiled objects with a separate wrapper class. As shown in Figure 8, the wrapper class "intercepts" calls to the object in order to allow it to function as a distributed object. Thus, even if distributed objects could be considered to be duplicated objects, which as discussed above is not the case, Snyder would not anticipate the independent claims.

Thus none of the cited references even suggests the possibility of program instructions that describe objects, which are to be shared by a plurality of network-connected terminals over a network, by means of ASCII instructions and also compile said ASCII instructions within an instructions set executable by said network-connected terminals, wherein said executable instructions set configures a processing means to manage the duplication of the objects.

Also, there can be no question of obviousness because the problems are so different. There is no reason why one of ordinary skill in the art should look to the field of distributed objects, which has at its heart the notion of accessing an object in a single location, for an answer to a problem with duplicating objects.

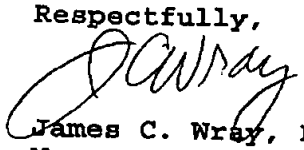
Any combination of Pham, Bannon, and Snyder would still not produce the invention. Since none of them even contemplate the duplicating of objects there is no reason why the combination of the three disclosures should lead one skilled in the art to suggest the present invention.

Therefore, all the independent claims are novel, inventive, and non-obvious over both Pham and Snyder, and also over the combination of them, since neither addresses the same or even a similar problem to the present application. Because the independent claims are patentable, the dependent claims are also allowable.

Nothing in the references, either singly or in combination, teaches or suggests the claimed features. Therefore, the references cannot anticipate nor render obvious the present invention as claimed.

Since Applicant has presented a novel, unique and non-obvious invention, reconsideration and allowance are respectfully requested.

Respectfully,



James C. Wray, Reg. No. 22,693
Meera P. Narasimhan, Reg. No. 40,252
1493 Chain Bridge Road, Suite 300
McLean, Virginia 22101
Tel: (703) 442-4800
Fax: (703) 448-7397

November 9, 2004